Express.js 빠른 참조

라우팅, 미들웨어, 요청, 응답, 패턴

설정

서버 생성 & 시작

```
const express = require("express");
const app = express();
app.listen(3000, () => console.log("Running on :3000"));
```

내장 미들웨어

```
app.use(express.json());          // parse JSON bodies
app.use(express.urlencoded({ extended: true })); // form data
app.use(express.static("public")); // serve static files
```

라우팅

HTTP 메서드

```
app.get("/users", (req, res) => res.json(users));
app.post("/users", (req, res) => res.status(201).json(req.body));
app.put("/users/:id", (req, res) => res.json(updated));
app.delete("/users/:id", (req, res) => res.sendStatus(204));
```

라우트 파라미터

```
app.get("/users/:id", (req, res) => {
  const { id } = req.params;
  res.json({ id });
});
```

쿼리 문자열

```
// GET /search?q=express&page=2
app.get("/search", (req, res) => {
  const { q, page } = req.query;
  res.json({ q, page });
});
```

미들웨어

애플리케이션 수준

```
app.use((req, res, next) => {
  console.log(`${req.method} ${req.url}`);
  next();
});
```

라우트 수준

```
const auth = (req, res, next) => {
  if (!req.headers.authorization) return res.sendStatus(401);
  next();
};
app.get("/secret", auth, (req, res) => res.json({ ok: true }));
```

실행 순서

app.use(fn)	모든 요청에 실행 (순서대로)
app.use(path, fn)	일치하는 경로 접두사에만 실행
next()	다음 미들웨어로 제어 전달
next(err)	에러 핸들러로 건너뜀

요청 & 응답

요청 객체

req.params	라우트 파라미터 (**/users/:id**)
req.query	쿼리 문자열 (**?key=val**)
req.body	파싱된 요청 본문 (파서 필요)
req.headers	요청 헤더 객체
req.method	HTTP 메서드 (GET, POST, ...)
req.path	URL 경로명
req.cookies	쿠키 (cookie-parser 필요)

응답 객체

res.json(obj)	JSON 응답 전송
res.send(body)	문자열/Buffer/객체 전송
res.status(code)	HTTP 상태 설정 (체이닝 가능)
res.redirect(url)	302 리다이렉트 (상태 코드 전달 가능)
res.sendFile(path)	파일을 응답으로 전송
res.sendStatus(code)	기본 텍스트와 함께 상태 전송
res.set(header, val)	응답 헤더 설정

에러 처리

에러 미들웨어

```
// Must have 4 parameters — Express recognizes it as error handler
app.use((err, req, res, next) => {
  console.error(err.stack);
  res.status(err.status || 500).json({ error: err.message });
});
```

에러 핸들러는 모든 app.use()와 라우트 뒤에 정의

비동기 에러

```
// Wrap async route handlers to catch rejections
const wrap = (fn) => (req, res, next) =>
  Promise.resolve(fn(req, res, next)).catch(next);

app.get("/data", wrap(async (req, res) => {
  const data = await fetchData();
  res.json(data);
}));
```

정적 파일

정적 디렉터리 제공

```
app.use(express.static("public"));
// serves public/style.css at /style.css

// With virtual path prefix
app.use("/assets", express.static("public"));
// serves public/style.css at /assets/style.css
```

옵션

dotfiles	**'ignore'** | **'allow'** | **'deny'**
maxAge	Cache-Control max-age (밀리초)
index	인덱스 파일명 (기본: **index.html**)
fallthrough	404 시 다음 미들웨어로 전달

템플릿

뷰 엔진 설정

```
app.set("view engine", "ejs");
app.set("views", "./views");

app.get("/", (req, res) => {
  res.render("index", { title: "Home", items: [1, 2, 3] });
});
```

주요 엔진

ejs	내장 JS 템플릿 (**<%= val %>**)
pug	들여쓰기 기반 (구 Jade)
handlebars	Mustache 스타일 (**{{val}}**)

라우터

모듈식 라우트

```
// routes/users.js
const router = require("express").Router();
router.get("/", (req, res) => res.json(users));
router.get("/:id", (req, res) => res.json(user));
module.exports = router;
```

라우터 마운트

```
const usersRouter = require("./routes/users");
app.use("/api/users", usersRouter);
// GET /api/users -> router's "/"
// GET /api/users/5 -> router's "/:id"
```

라우터 메서드

router.get/post/put/delete	HTTP 메서드 핸들러
router.use(fn)	라우터 수준 미들웨어
router.param(name, fn)	라우트 파라미터 전처리
router.route(path)	한 경로에 메서드 체이닝

인증 패턴

JWT 미들웨어

```
const jwt = require("jsonwebtoken");
const auth = (req, res, next) => {
  const token = req.headers.authorization?.split(" ")[1];
  if (!token) return res.sendStatus(401);
  req.user = jwt.verify(token, process.env.SECRET);
  next();
};
```

보호된 라우트

```
app.get("/profile", auth, (req, res) => {
  res.json({ user: req.user });
});
app.use("/api/admin", auth, adminRouter);
```

일반 패턴

CORS

```
const cors = require("cors");
app.use(cors());                          // allow all origins
app.use(cors({ origin: "https://example.com" })); // restrict
```

환경 & 설정

```
const port = process.env.PORT || 3000;
app.listen(port);

// Access env in routes
if (app.get("env") === "production") {
  app.use(helmet());
}
```

유용한 npm 패키지

cors	크로스 오리진 리소스 공유
helmet	보안 헤더
morgan	HTTP 요청 로거
cookie-parser	Cookie 헤더 파싱
dotenv	**.env**를 **process.env**에 로드
multer	멀티파트 폼 데이터 (파일 업로드)